Docket No. P8526 NP/US
Firm No. 0045.85

METHOD, SYSTEM, AND PROGRAM FOR CHARACTER SET MATCHING

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] The present invention is related to matching a character set against character sets in one or more data set files.

2. Description of the Related Art

[0002] Often times, individuals search through a large amount of data trying to find particular information. Many search engines enable an individual to enter one or more keywords that may be used to locate the desired information. The search engine, however, may take a long time in its attempt to match keywords to the large amount of data.

[0003] There are several situations in which it is highly desirable to obtain a quick result when a large population of data is to be searched to identify a match for a set of keywords. For example, consider the general case of an on-line purchasing process via, for example, a Web site on the Internet. The Internet is a world-wide collection of connected computer networks (i.e., a network of networks). The World Wide Web (also referred to as the "Web") is a system of Internet servers that support documents (e.g., Web pages). A Web site refers to a location on the Web at which one or more documents (e.g., one or more Web pages) are displayed. A buyer typically views a Web site of a company. The buyer has a set of criteria represented by keywords to be used to scan products for sale by the company on the Web site. A product with the maximum number of matches to the criteria is most likely to be the product that the buyer is looking for. For example, the buyer may search for a "blue sweater." A product that is identified as "blue" and as a "sweater" would be identified as the best match for the criteria.

[0004] Another example in which keywords may be used to locate particular information is in the hiring process. In this case, an employer may post job requirements, which

includes a set of criteria represented by keywords (e.g., skill set, level of education, experience, etc.) that an applicant should possess in order to be able to fill the job requirements. The set of resumes for all applicants is scanned, looking for matches to the desired keywords in the job listing. Then, the employer may review the applicants who appear to be qualified for the job.

[0005] The problem of having to scan a large amount of data to select elements with properties that match a given set of keywords exists in many fields. However, existing search engines are oftentimes slow. Thus, there is a need for improved search techniques.

SUMMARY OF THE INVENTION

[0006] Provided are a method, system, and program for matching character sets. One or more data set files are stored. One or more character set files are created, wherein each character set file is associated with a character set and includes indexes associated with the one or more data set files. A request specifying one or more character sets is received. One or more of the data set files that contain one or more of the requested character sets are identified using the indexes in the character set files.

[0007] The described implementations of the invention provide a method, system, and program for providing both a technique for structuring data and a technique for character set matching that scans the data to provide a set of matches.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates, in a block diagram, a computing environment in accordance with certain implementations of the invention.

FIG. 2 illustrates, in a block diagram, data organization in accordance with certain implementations of the invention.

FIG. 3 illustrates logic implemented in accordance with certain implementations of the invention.

FIGs. 4A, 4B, and 4C illustrate logic implemented in the character set matching system in accordance with certain implementations of the invention.

FIGs. 5A, 5B, 5C, 5D, 5E, and 5F illustrate arrays of index entries and output files used in describing an example in accordance with certain implementations of the invention.

FIG. 6 illustrates an architecture of a computer system that may be used in accordance with certain implementations of the invention.

DETAILED DESCRIPTION

[0008] In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several implementations of the present invention. It is understood that other implementations may be utilized and structural and operational changes may be made without departing from the scope of the present invention.

[0009] FIG. 1 illustrates, in a block diagram, a computing environment in accordance with certain implementations of the invention. A client computer 100 executes one or more client applications 110. A client application 110 may be any type of application program. The client computer 100 is connected to a server computer 120 by a network 190, such as a local area network (LAN), wide area network (WAN), or the Internet. The Internet is a world-wide collection of connected computer networks (i.e., a network of networks). The computer applications 110 may access data at the server computer 120.

[0010] The server computer 120 includes a character set matching system 130 and one or more server applications 140. The server applications 140 may be any type of applications. The character set matching system 130 receives a request to match a character set against a set of data set files located in one or more data set directories and identifies zero or more data set files that include the character set. Moreover, the server

computer 120 is connected to storage devices 160, 170, and each storage device 160, 170 has a device interface 162, 172. For example, each storage device 160 and 170 may be a redundant array of independent disks (RAID). A RAID device enables storage of the same data on multiple hard disks, thus allowing simultaneous accesses to copies of the data.

[0011] Implementations of the invention structure data in a manner that enables improved character set matching. A character set comprises one or more characters, including, for example, alphabet letters, numbers, blank spaces, and symbols (e.g., *, #, &).

[0012] In terms of data organization, implementations of the invention construct data in the form of data set files (e.g., data files). Data set files may contain information in a "structured" or a "free form" format and include certain character sets (e.g., words) that are regarded as "keywords" that are of importance to a given application domain. The term "structured" format refers to data being organized in a particular manner. For example, a book is said to have a "structured" format since a book is organized in terms of certain rules, such as, chapters, paragraphs, indexes, etc. The term "free form" format refers to the fact that the data set files may be unstructured and need not contain data organized in any particular manner or according to any predefined rules.

[0013] FIG. 2 illustrates, in a block diagram, data organization in accordance with certain implementations of the invention. In particular, FIG. 2 illustrates a data set directory 210, a character set directory 220, and relationships between data set files in the data set directory 210 and character set files in the character set directory 220. Each character set file (e.g., 222, 224, 226) contains one or more indexes (or pointers) to various data set files (in the data set) that contain one or more instances of the corresponding character set (i.e. keyword). Each data set file (e.g., 212, 214, 216) contains data to be matched to the character sets. The data set files contain structured or unstructured data (e.g., words). In certain implementations, the data belongs to a specific application domain of interest, e.g. resume files, job requirement files, product description files, etc.

[0014] Each data set file is identified in the system by a unique index assigned to the data set file when the data set file is created. The index may be any unique identifier that indicates the location of the data set file. In certain implementations, the indexes can be simply the absolute path name of the data set files or can be Extensible Markup Language (XML) pointers.

[0015] Character sets for which character set files are to be created may be identified by, for example, a system administrator or other individual. For each identified character set, implementations of the invention maintain an associated character set file. For example if the character sets: programming, user interfaces, object-oriented, and management are the desirable character sets, then four character set files are maintained, one file for each one of the character sets. Character set files contain indexes to individual data set files, which contain one or more instances of the character set associated with the character set file occurring one or more times. The indexes in a character set file are maintained in sorted order.

[0016] Whenever a new data set file is added to the data set directory 210, the new data set file is assigned an index and is scanned for character sets associated with the character set files in the character set directory 220. For each character set that appears one or more times in the data set file, the index of the data set file is inserted in sorted order in the associated character set file. For example, character set file-1 222 includes indexes to data set file 212 and data set file 214, and character set file-2 224 includes indexes to data set file 212 and data set file 216.

[0017] FIG. 3 illustrates logic implemented in accordance with certain implementations of the invention. Control begins at block 300 with data being organized, for example, by a system administrator or other individual, as a data set directory of data set files. In block 310, character sets are identified, by, for example, a system administrator or other individual. In certain implementations of the invention, the character sets may be identified with computer software or hardware that is implemented to consider various factors. For example, data set files may be structured in accordance with certain rules,

such as those of XML, in which case character sets are constructed in accordance with specific rules and formats, which enables a computer program to parse data set files and automatically identify desired character sets. In block 320, a character set file is created for each identified character set. In block 330, for each data set in the data set directory, an index to that data set file is stored in a character set file if the character set associated with the character set file is found in the data set file.

[0018] FIGs. 4A, 4B, and 4C illustrate logic implemented in the character set matching system 130 in accordance with certain implementations of the invention. Control begins at block 400 with the character set matching system 130 creating an array of index entries by reading a single index from each of the character set files. In block 402, the character set matching system 130 selects the first index entry in the array. The selected index entry is also referred to as a "current index entry."

[0019] In block 404, the character set matching system 130 determines whether all index entries in the array have been selected. If so, processing continues to block 421 (FIG. 4B), otherwise, processing continues to block 406. In block 406, the character set matching system 130 selects another index entry whose index is to be compared with the index stored in the current index entry, and this other index entry is referred to as a "compared index entry."

[0020] In certain implementations of the invention, the indexes are stored in character set files in non-numeric format (e.g., as XML paths) and are converted to a numeric format for comparison. In certain alternative implementations of the invention, the indexes are stored in the character set files in a numeric format in ascending order (e.g., 1, 2, 3, ...). In block 408, the character set matching system 130 determines whether the index stored in the current index entry is less than the index stored in the compared index entry. That is, the character set matching system 130 compares the index stored in the current index entry to each index stored in each other index entry, one by one. If the index stored in the current index entry is less than the index stored in the compared index entry, processing loops back to block 406, otherwise, processing continues to block 410.

[0021] In block 410, the character set matching system 130 determines whether the index stored in the current index entry is equal to the index stored in the compared index entry. If so, processing continues to block 412, otherwise, processing continues to block 420. In block 412, the character set matching system 130 increments a matching counter. In block 414, the character set matching system 130 sets an update indicator (e.g., a flag) to update the index stored in the compared index entry with the next index from the character set file corresponding to the compared index entry. For example, if a character set file has index-1 and index-2 and the compared index entry stores index-1, then an update indicator is set to update the compared index entry with index-2 for another iteration of processing. Note that the update occurs in block 426. After the processing of block 414, processing loops back to block 404.

[0022] In block 420, the character set matching system 130 sets the compared index entry to be the current index entry and resets update indicators and the matching counter. The update indicators would have been set for index entries of a particular value that is greater than the compared index entry, but, if the current index entry is changed to the compared index entry, then the update indicators are reset so that the index entries are not updated.

[0023] When the index stored in the current index entry is greater than the index stored in the compared index entry, then the compared index entry becomes the current index entry and processing continues by comparing the index stored in the new current index entry with the indexes stored in the remaining index entries. If a character set file has no more indexes, then the index entry in the array 500 corresponding to the character set file is set to, for example, NULL or some other value, to indicate that there are no more indexes in the character set file, and, hence will be skipped during the following iterations of comparisons. Additionally, in certain implementations of the invention, selection of an index entry selects a non-NULL index entry.

[0024] In block 421, the character set matching system 130 increments the matching counter. In block 422, the character set matching system 130 writes the index stored in the current index entry in the output file associated with the value of the matching

counter. That is, each matching counter value has an associated output file for storing matching index entries. The number of output files is equal to the number of given character sets, and, in certain implementations of the invention, the output files may be identified as: outputfile1, outputfile2,..., outputfilek, where k = number of given character sets. For example, outputfile1 contains indexes for data set files that include one of the given character sets, and outputfile2 contains indexes for data set files that include two of the given character sets, and so on.

[0025] In block 424, the character set matching system 130 determines whether all index entries in the character set files have been selected. If so, processing loops to block 428, otherwise, processing loops back to block 426. In block 426, the character set matching system 130 updates the current index entry by another index from the character set file corresponding to the current index entry and containing the current index and updates any index entry whose update indicator is set to indicate that the index entry is to be updated, and loops back to block 404.

[0026] In block 428, the results are output. The results include outputfilek with all indexes to data set files in the data set directory that match all k character sets, followed by outputfilek-1 with matches to k-1 character sets, outputfilek-2,..., outputfile2, outputfile1.

[0027] FIGs. 5A, 5B, 5C, 5D, 5E, and 5F illustrate arrays of index entries and output files used in describing an example in accordance with certain implementations of the invention. Each index entry stores an index from a corresponding character set file. Each output file stores zero or more indexes. For this example, there are four character set files. Each character set file has a column of indexes that are represented by numbers and which are ordered in ascending order. In particular, a first character set file includes indexes: 5, 10. A second character set file includes indexes: 3, 10. A third character set file includes index: 20. A fourth character set file includes indexes: 2, 5, 10. Although the example described herein is very simple to assist with understanding of

implementations of the invention, implementations of the invention may work with many more character set files containing many more indexes.

[0028] Initially, an index from each of the four character set files is selected and stored in a corresponding index entry in an array of index entries 500 (FIG. 5A). Index entry 502 contains five, index entry 504 contains three, index entry 506 contains twenty, and index entry 508 contains two. The first ("current") index entry 502 of five is compared to the next index entry 504 of three. Since the current index entry 502 of five is greater than the compared index entry 504 of three, the processing in block 420 occurs. That is, the current index entry is set to index entry 504 of three and any update indicators are reset. In this example, there are no update indicators set at this time.

[0029] The next index entry 506 of twenty is selected. Since the current index entry 504 of three is less than the compared index entry 506 of twenty, the next index entry 508 of two is selected for comparison. Since the current index entry 504 of three is greater than the compared index entry 508 of two, the compared index entry 508 of two becomes the current index entry.

[0030] Since all index entries in the array 500 have been selected, a matching counter for two is incremented to indicate that there is one match, and the index of two is stored in outputfile1 552, which stores the indexes whose matching counter is one.

[0031] Also, since there are additional index entries in the fourth character set file of this example, index entry 508 is updated with another index entry in the character set file that contains index entry 508. In this example, FIG. 5B illustrates an array of index entries 510 in which index entry 508 has been updated to contain five.

[0032] For the next iteration, index entry 502 is the current index entry. Since the current index entry 502 of five is greater than index entry 504 of three, index entry 504 becomes the current index entry. Since the index entry 504 of three is less than index entry 506 of twenty and index entry 508 of five, and all index entries in the array 510 have been selected (block 404), processing continues to block 421. The index of three is stored in outputfile1 552.

[0033] FIG. 5C illustrates an array of index entries 520 in which index entry 504 is updated to ten. For the next iteration, the current index entry is index entry 502 of five. In this iteration, index entry 502 of five is found to match index entry 508 of five (in block 410). Therefore, the matching counter is incremented to one, indicating there is one match for index entry 502 of five. An update indicator is set for index entry 508 of five. Then processing loops back to block 404. Since all index entries in the array 520 have been selected, processing continues to block 421. The matching counter is incremented to two, and the index of five is stored in outputfile2 554, which contains index entries with two matches. Then, the current index entry 502 of five and the index entry 508, whose update indicator is set, are updated.

[0034] FIG. 5D illustrates an array of index entries 530 in which index entry 502 and index entry 508 have been updated to ten. In this iteration, it is determined that there are three matches for index entry of ten, and the index of ten is stored in outputfile3 556.

[0035] FIG. 5E illustrates an array of index entries 540 in which index entry 502, index entry 504, and index entry 508 have been updated to NULL to indicate that there are no more index entries in the character set files associated with these index entries in the array. Then, the index of twenty is stored in outputfile1 552.

[0036] FIG. 5F illustrates outputfile1 552, outputfile2 554, outputfile3 556, and outputfile4 558, which would be output at block 428.

[0037] Table A lists pseudocode written in the JAVA™ programming language that represents processing performed by the matching technique in accordance with certain implementations of the invention.

Table A

```
import java.io.*;
public class Match {
public static void main(String args[]) {
int nTokens = args.length;
```

```
String href [] = new String[nTokens];
boolean hrefUpdate [] = new boolean[nTokens];
TokenFile tf [] = new TokenFile[nTokens];
OutFile of [] = new OutFile[nTokens];
boolean more_to_compare = false;
String baseDIR = "<base directory of keyword files>";
String current_value = null;
String tmp_value = null;
int hrefPtr = 0;
int count = 0;
int i;
// initialize the token & output files & fill up the href array.
for (i=0; i<nTokens; i++) {
try {
tf[i] = new TokenFile(baseDIR + args[i]);
href[i] = tf[i].readLine();
hrefUpdate[i] = false;
of[i] = null;
} catch (IOException e) {
System.out.println("couldn't read token file");
}
}
do {
hrefPtr = 0;
count = 0;
i = 0;
current_value = null;
for (i=0; i<nTokens; i++) {
```

```
if (current_value == null) {
if (href[i] != null) {
current_value = href[i];
hrefUpdate[i] = true;
count = 0;
tmp_value = null;
continue;
}
else
continue;
}
// at this point current_value is not null.
// if tmp_value is null then skip this iteration.
tmp_value = href[i];
if (tmp_value == null)
continue;
int cmp = current_value.compareTo(tmp_value);
if (cmp < 0) {
continue;
}
else if (cmp == 0) {
count++;
hrefUpdate[i] = true;
}
else { // i.e. last case: current_value > tmp_value.
// clear hrefUpdate array
for (int j=0; j<nTokens; j++)
hrefUpdate[j] = false;
```

```
current_value = tmp_value;
hrefUpdate[i] = true;
count = 0;
}
} // end of for.
// if we got current_value != null, then need to output it
// into one of the output file according to its count.
// need also to update the cells of hrefUpdate by reading
// a new line from the token file
if (current_value != null) {
String ofile = String.valueOf(count + 1);
// check if this is the first time we create
// OutFile object for this file.
if (of[count] == null)
of[count] = new OutFile(ofile);
// write down current_value in the output file
try {
of[count].writeLine(current_value);
} catch (IOException e) {
System.out.println("couldn't write to output file");
}
// update "current_value"(s) with new ones from their
// token file.
for (int j=0; j<nTokens; j++) {
if (hrefUpdate[j] == true) {
try {
href[j] = tf[j].readLine();
hrefUpdate[j] = false;
```

```
} catch (IOException e) {
System.out.println("couldn't write to output file");
}
}
}
} // end of if
} while (current_value != null);
}
}
```

[0038] The matching technique sweeps through each of the character set files only once in a concurrent fashion to read index entries. Each index entry value is then compared to other index entry values. The possible number of comparison operations for each index value ranges from zero up to k-1 operations. A worst case scenario may occur if each character set file contain an exclusive set of indexes. To calculate complexity, assume n to be the total number of all indexes in all character set files, and that each file has the same number of indexes (i.e. n/k indexes, where k is the number of the given character sets). Therefore: $O(n) = (k-1)(n/k) + (k-2)(n/k) + ...+ (2)(n/k) + (n/k)$

$= (k(k-1))/2 * (n/k)$

$= (k-1)/2 * n$

Although the above formula represents a worst case scenario, in typical applications, k is really much less than n, which makes the complexity close to linear.

[0039] The matching technique produces a set of output files that include the indexes to data set files with matches to the given set of character sets. For example, in the case of searching for candidates with skill sets that match a given job listing, a search may be performed for candidates with four skills (e.g., programming, user interfaces, object-

oriented, and management). The matching technique finds the data set files (e.g., resume files) with matches to one or more of the four character sets.

[0040] The described techniques for character set matching may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The term "article of manufacture" as used herein refers to code or logic implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.) or a computer readable medium, such as magnetic storage medium (e.g., hard disk drives, floppy disks,, tape, etc.), optical storage (CD-ROMs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, firmware, programmable logic, etc.). Code in the computer readable medium is accessed and executed by a processor. The code in which embodiments are implemented may further be accessible through a transmission media or from a file server over a network. In such cases, the article of manufacture in which the code is implemented may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, infrared signals, etc. Thus, the "article of manufacture" may comprise the medium in which the code is embodied. Additionally, the "article of manufacture" may comprise a combination of hardware and software components in which the code is embodied, processed, and executed. Of course, those skilled in the art will recognize that many modifications may be made to this configuration without departing from the scope of the present invention, and that the article of manufacture may comprise any information bearing medium known in the art.

[0041] The logic of FIGs. 3 and 4A-4C describes specific operations occurring in a particular order. In alternative implementations, certain of the logic operations may be performed in a different order, modified or removed. Moreover, steps may be added to the above described logic and still conform to the described implementations. Further, operations described herein may occur sequentially or certain operations may be

processed in parallel, or operations described as performed by a single process may be performed by distributed processes.

[0042] The illustrated logic of FIGs. 3 and 4A-4C was described as being implemented in software. The logic may be implemented in hardware or in programmable and non-programmable gate array logic.

[0043] FIG. 6 illustrates an architecture of a computer system 100, 120 that may be used in accordance with certain implementations of the invention. The computer architecture 600 may implement a processor 602 (e.g., a microprocessor), a memory 604 (e.g., a volatile memory device), and storage 610 (e.g., a non-volatile storage area, such as magnetic disk drives, optical disk drives, a tape drive, etc.). An operating system 605 may execute in memory 604. The storage 610 may comprise an internal storage device or an attached or network accessible storage. Computer programs 606 in storage 610 may be loaded into the memory 604 and executed by the processor 602 in a manner known in the art. The architecture further includes a network card 608 to enable communication with a network. An input device 612 is used to provide user input to the processor 602, and may include a keyboard, mouse, pen-stylus, microphone, touch sensitive display screen, or any other activation or input mechanism known in the art. An output device 614 is capable of rendering information transmitted from the processor 602, or other component, such as a display monitor, printer, storage, etc. The computer architecture 600 of the computer systems may include fewer components than illustrated, additional components not illustrated herein, or some combination of the components illustrated and additional components.

[0044] The computer architecture 600 may comprise any computing device known in the art, such as a mainframe, server, personal computer, workstation, laptop, handheld computer, telephony device, network appliance, virtualization device, storage controller, etc. Any processor 602 and operating system 605 known in the art may be used.

[0045] The foregoing description of implementations of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to

limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many implementations of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

[0046] JAVA is a registered trademark or trademark of Sun Microsystems, Inc. in the United States and/or other countries.